8

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/966,185	09/28/2001	Hoi Lee Candy Wong	10745/026	5520

27879 7590 05/07/2004

INDIANAPOLIS OFFICE 27879
BRINKS HOFER GILSON & LIONE
ONE INDIANA SQUARE, SUITE 1600
INDIANAPOLIS, IN 46204-2033

EXAMINER	
PESIN, BORIS M	
ART UNIT	PAPER NUMBER
2174	3

DATE MAILED: 05/07/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/966,185	**Applicant(s)** WONG ET AL.	
	Examiner Boris Pesin	**Art Unit** 2174	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)□ Responsive to communication(s) filed on _____.

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)□ Claim(s) 1-30 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)□ Claim(s) 1-30 is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) □ Notice of Informal Patent Application (PTO-152)

6) □ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-15 and 17-30 are rejected under 35 U.S.C. 102(b) as being anticipated by Abrams, Marc et al. ("UIML: An XML Language for Building Device-Independent User Interfaces").

In regards to claim 1, Abrams teaches, a method of scaling an application graphical user interface for display with any of a plurality of heterogeneous device platforms, the method comprising:

a) creating an instance of an intermediate representation of a device platform independent application graphical user interface (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts* *and events. These names are mnemonics for the interface implementor."* Page 5);

b) dynamically customizing the intermediate representation as a function of the capabilities of one of the heterogeneous device platforms (i.e. "*A UIML document can be used in one of several ways. First, it can be stored on a server (see Figure 1). When a user invokes an application, the UIML document either is compiled to a target platform's language (e.g., to WML, or to C++), or it is interpreted while the user interacts with the interface.*" Page 3);

c) extracting a device platform dependent application graphical user interface from the customized intermediate representation (i.e. "*Compilation on the server side is mandatory for devices that cannot download applications, such as cellular phones, and where memory is limited. Interpretation is more flexible; for example our Java interpretive renderer permits the entire UIML interface to appear as a Java bean, so that the interface may be manipulated programmatically by the application logic.*" Page 3); and

d) displaying the device platform dependent application graphical user interface with said one of the heterogeneous device platforms (i.e. Figure 1, Page 4).

In regards to claim 2, Abrams teaches a method dependent on claim 1, wherein b) comprises removing unnecessary tasks from the intermediate representation as a function of the capabilities of said one of the heterogeneous device platforms. (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An* *interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11).

In regards to claim 3, Abrams teaches a method dependent on claim 1, wherein b) comprises transforming the intermediate representation as a function of the capabilities of said one of the heterogeneous device platforms (i.e. "*Dynamic Interfaces are interfaces that are generated on-the-fly and are usually custom-made for each user or device. There are two reasons why dynamic interfaces are becoming very popular. First there is a plethora of devices available in the market today and users maybe accessing an application with anyone of them.*" Page 10).

In regards to claim 4, Abrams teaches a method dependent on claim 1, wherein a) comprises migrating a scalable application to a mobile device, the scaleable application comprising the device platform independent application graphical user interface (Figure 1, Page 4, Device #2).

In regards to claim 5, Abrams teaches a method dependent on claim 1, wherein a) comprises representing at least one device independent graphical user interface component with the intermediate representation (i.e. "*Dynamic Interfaces are interfaces that are generated on-the-fly and are usually custom-made for each user or device. There are two reasons why dynamic interfaces are becoming very popular. First there is a plethora of devices available in the market today and users maybe accessing an application with anyone of them.*" Page 10).

In regards to claim 6, Abrams teaches a method dependent on claim 1, wherein a) comprises instantiating the intermediate representation with a server computer (i.e. "*A UIML document can be used in one of several ways. First, it can be stored on a server (see Figure 1). When a user invokes an application, the UIML document either is compiled to a target platform's language (e.g., to WML, or to C++), or it is interpreted while the user interacts with the interface.*" Page 3).

In regards to claim 7, Abrams teaches a method dependent on claim 1, wherein a) comprises transmitting the intermediate representation to a server computer (i.e. "Figure 1, Page 4, "User Profile").

In regards to claim 8, Abrams teaches a method dependent on claim 7, wherein b) comprises transmitting the customized intermediate representation from the server computer to said one of the heterogeneous device platforms (i.e. "Figure 1, Page 4, "UIML and WML").

In regards to claim 9, Abrams teaches a method dependent on claim 1, wherein a) comprises hierarchically representing a plurality of tasks with the intermediate representation (i.e. "*The enumeration of parts is given in a hierarchical form, to designate the logical structure of the interface.*" Page 4).

In regards to claim 10, Abrams teaches a method of scaling an application graphical user interface for display on a display screen of any of a plurality of heterogeneous device platforms, the method comprising:

a) instantiating a logic structure from a device platform independent application graphical user interface, the logic structure comprising at least one logical panel and representation of at least one graphical user interface component interface (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts and events. These names are mnemonics for the interface implementor.*" Page 5);

b) selectively retaining the at least one logical panel and the at least one graphical user interface component in the logic structure as a function of suitability of the at least one logical panel and the at least one graphical user interface component to one of the heterogeneous device platforms (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11);

c) configuring the at least one logical panel and the at least one graphical user interface component in a layout compatible with the capabilities of a display screen of said one of the heterogeneous device platforms (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11); and

d) extracting the at least one graphical user interface component and the at least one logical panel from the logic structure (i.e. "*Compilation on the server side is mandatory for devices that cannot download applications, such as cellular phones, and where memory is limited. Interpretation is more flexible; for example our Java interpretive renderer permits the entire UIML interface to appear as a Java bean, so that the interface may be manipulated programmatically by the application logic.*" Page 3).

In regards to claim 11, Abrams teaches a method dependent on claim 10, further comprising e) displaying a device platform dependent application graphical user interface on the display screen as a function of the at least one graphical user interface component and the at least one logical panel extracted from the logic structure (i.e. Figure 1, Page 4).

In regards to claim 12, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying properties comprising layout parameters of the logic structure with the device independent application graphical user interface interface (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts and events. These names are mnemonics for the interface implementor.*" Page 5).

In regards to claim 13, Abrams teaches a method dependent on claim 10, wherein b) comprises removing the at least one graphical user interface component from the logic structure as a function of tasks unsuitable to said one of the heterogeneous device platforms (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11).

In regards to claim 14, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying retention of the at least one logical panel within the logic structure with the device platform independent application graphical user interface (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An* *interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11).

In regards to claim 15, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying a suggested layout structure for the at least one logical panel and the at least one graphical user interface component with the device platform independent application graphical user interface (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts and events. These names are mnemonics for the interface implementor.*" Page 5).

In regards to claim 17, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying layout constraints for the at least one graphical user interface component with the device platform independent application graphical user interface (i.e. "*Appendix. A Complete UIML Example Figure 4 shows two possible renderings of the credit card template. On the left is a Java-AWT dialog and on the right is a WML card.*" Pages 12-15).

In regards to claim 18, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying layout groups of the at least one logical panel with the device platform independent application graphical user interface (i.e. "*Appendix. A Complete UIML Example Figure 4 shows two possible renderings of the credit card template. On the left is a Java-AWT dialog and on the right is a WML card."* Pages 12-15).

In regards to claim 19, Abrams teaches a method dependent on claim 10, wherein a) comprises specifying a label for the at least one logical panel with the device platform independent application graphical user interface (i.e. "*Appendix. A Complete UIML Example Figure 4 shows two possible renderings of the credit card template. On the left is a Java-AWT dialog and on the right is a WML card.",* Pages 12-15, especially the code).

In regards to claim 20, Abrams teaches a method dependent on claim 10, wherein d) comprises accessing a scalable graphical user interface library to obtain the at least one graphical user interface component (i.e. "*This paper highlights two aspects of UIML: the ability to generate user interfaces for different platforms (we give example for Java AWT, WML, and VoiceXML), and the ability to construct a library of reusable interface components.*" Page 1).

In regards to claim 21, Abrams teaches a method dependent on claim 10, wherein d) comprises downloading the at least one graphical user interface component from a server computer (i.e. "Figure 1, Page 4, "UIML and WML")..

In regards to claim 22, Abrams teaches a scalable graphical user interface architecture for scaling an application graphical user interface to the display screen of any of a plurality of heterogeneous device platforms, the scalable graphical user interface architecture comprising: a target device platform comprising a display screen (inherent in Abrams); a device platform independent application graphical user interface operable within the target device platform to initiate creation of an instance of an intermediate representation of the device platform independent application graphical user interface (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts and events. These names are mnemonics for the interface implementor.*" Page 5); a customizing module operable to customize the intermediate representation as a function of the capabilities of the target device platform (Figure 1, Page 4, "Interface Server"); and a render manager module operable to produce a device platform dependent application graphical user interface on the display screen as a function of the customized intermediate representation (Figure 1, Page 4, "Renderer").

In regards to claim 23, Abrams teaches a scalable graphical user interface architecture, further comprising a scalable graphical user interface library operable with the device platform independent application graphical user interface to create the instance of the intermediate representation (i.e. "*This paper highlights two aspects of UIML: the ability to generate user interfaces for different platforms (we give example for Java AWT, WML, and VoiceXML), and the ability to construct a library of reusable interface components.*" Page 1).

In regards to claim 24, Abrams teaches a scalable graphical user interface architecture, wherein the customizing module comprises a task manager module operable to retain tasks applicable to the functionality of the target: device platform and the display screen (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies supported by the device without overloading the network or the device with unsupported code.*" Page 11).

In regards to claim 25, Abrams teaches a scalable graphical user interface architecture, wherein the customizing module comprises a transformation manager module operable to configure the layout structure of the display as a function of properties specified by the device platform independent application graphical user interface (i.e. "*Second, UIML can be dynamically generated from a database (see Figure 3). An interface server can query the client for a list of device features and/or user preferences and dynamically generate the UIML code for the interface from a database query. The user gets a tailored UI that takes advantage of the technologies* *supported by the device without overloading the network or the device with unsupported code.*" Page 11, Also See Figure 1, Page 4).

In regards to claim 26, Abrams teaches a scalable graphical user interface, wherein the scalable graphical user interface library comprises a user interface event translator module, the user interface event translator module operable to translate a graphical user interface event generated by the target device platform to an action compatible with the device platform independent application graphical user interface (i.e. See Figure 1, Page 4, the renderer translates the general GUI into a GUI customized for the specific device).

In regards to claim 27, Abrams teaches a scalable graphical user interface architecture, wherein the intermediate representation comprises at least one logical panel and representation of at least one graphical user interface component (i.e. "*UIML can be viewed as a meta- or extensible language, analogous to XML. UIML does not contain tags specific to a particular user interface toolkit (e.g., <WINDOW> or <MENU>). Instead, it uses a set of generic tags (e.g., <part>, <property>). As discussed earlier, UIML captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occur for those parts, presentation style, content, and interconnection to application logic. The UIML author specifies instance and class names of their own choice for interface parts and events. These names are mnemonics for the interface implementor.*" Page 5).

In regards to claim 28, Abrams teaches a scalable graphical user interface architecture, wherein the target device platform comprises one of a pager, a wireless phone, a personal digital assistant, a hand-held personal computer, a vehicle navigation system and a notebook personal computer (Figure 1, Page 4, Device #1 and Device #2).

In regards to claim 29, Abrams teaches a scalable graphical user interface architecture, wherein the device platform independent application graphical user interface is compatible with, but device platform independent of any of the heterogeneous device platforms (i.e. "*An objective of UIML is to permit a UIML document to be mapped to any type of user interface, from graphical to speech, and even to those not yet invented. UIML represents an interface in five parts: the interface structure, presentation style, content (words, images, sounds), actions taken in response to user interaction, and interconnection of the interface to application logic.*" Page 1).

In regards to claim 30, Abrams teaches a scalable graphical user interface architecture wherein the target device platform comprises any, one of a plurality of heterogeneous device platforms (i.e. "*An objective of UIML is to permit a UIML document to be mapped to any type of user interface, from graphical to speech, and even to those not yet invented. UIML represents an interface in five parts: the interface structure, presentation style, content (words, images, sounds), actions taken in response to user interaction, and interconnection of the interface to application logic.*" Page 1).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Abrams, Marc et al. ("UIML: An XML Language for Building Device-Independent User Interfaces") in view of Shirakawa (US 5956738).

In regards to claim 16, Abrams teaches all the limitations of claim 10. He does not teach a method comprising specifying a layout priority for the at least one logical panel with the device platform independent application graphical user interface. Shirakawa teaches, "*The allocation section 106 refers to the article layout order 125 read out from the layout data group storage 105 and checks whether the article with the highest layout priority among the articles not yet allocated to any column can be selected as the processing subject or not (Step 1201).*" It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Abrams with the teachings of Shirakawa and include a method of specifying the layout priority with the motivation to better display the most important features on the device.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

US005583983A Schmitter

Teaches a multi-platform object-oriented software development and deployment system.

US006353448B1 Scarborough et al.

Teaches development and validation of multi-platform and multi-standard website layouts, this embodiment comprises the steps of storing a plurality of rendering algorithms each associated with a predefined graphic user interface standard, accessing a data repository storing an instruction set for constructing a graphic user interface layout through a network connection, rendering an image of the graphic user interface layout associated with the instruction set for each of the plurality of rendering algorithms, reducing the image of the graphic user interface layout to a bitmap image, and displaying the bitmap image for each of the plurality of rendering algorithms.

Farooq, Ali. "Simplifying Construction of Multi-Platform User Interfaces Using UIML" March 2001

Inquiry

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Boris Pesin whose telephone number is (703) 305-8774. The examiner can normally be reached on Monday-Friday except every other Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kristine Kincaid can be reached on (703) 308-0640. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Kristine Kincaid
KRISTINE KINCAID
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100